IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	A. Christian Tahan	Examiner:	Robert W. Morgan
Application No:	09/784,751	Art Unit:	3626
Filing Date:	02/15/2001		
Title:	Method Of Using a Global Server For Providing Patient Medical Histories To Assist In the Delivery Of Emergency Medical Services		
Atty. Docket:	XWRLD-102		

THIRD SUPPLEMENTAL RULE 131 DECLARATION

Commissioner of Patents & Trademarks
U.S. Patent and Trademark Office
P. O. Box 1450
Alexandria, VA 22313-1450

Now comes A. Christian Tahan and deposes and says:

1. That I am submitting this third Rule 131 Declaration to submit evidence that prior to February 22, 2000 I actually reduced to practice the claimed invention on at least on two occasions.

2. That prior to February 22, 2000 the first reduction to practice was at my father's house at 1312 Edward Drive, Moncks Corner, South Carolina, in which a database was populated with patient information, patient identification was inputted, the information about the patient was inputted from a remote site and the database was queried to provide information back to a wireless Palm device for emergency personnel at an accident scene based on identification of the patient.

3. That Appendix A is Samir Tahan's Declaration in Support attesting to the above

4. That prior to February 22, 2000 this first actual practice took place on a computer at 1312 Edward Drive which had the required data entry devices, including a keyboard and had requisite display for displaying the results of the query of the database which were transmitted to a wireless Palm device.

5. That Appendix B provides photographs of the server used at 1312 Edward Drive for the actual reduction to practice of the claimed invention prior to February 22, 2000; and a series of screen shots which were the result of running the Corel program on the server at 1312 Edward Drive prior to February 22, 2000 that indicates that Windows 98 was the operating system, that Netscape Communicator was the browser utilized, that an actual patient record was stored on the server, that a login form for Quest Rx was used to enter information into the database, that a specific ID was assigned to the patient and that the Corel database was used for storage of patient information.

6. That Appendix C is a photograph of the Palm handheld device used in the actual reduction to practice at 1312 Edward Drive prior to February 22, 2000.

7. That this reduction to practice was witnessed by Samir Tahan.

8. That the second actual reduction to practice occurred prior to February 22, 2000 on my personal computer at MIT witnessed by Alexandra Dunn who was familiar with my work.

9. That prior to February 22, 2000 Alexandra Dunn at a demonstration of the claimed system at MIT saw me populate a computer database on my personal computer with patient information by filling out a form and entering a Patient ID into a database in my personal computer that had patient records, that she saw me retrieve the relevant

patient information which was transmitted to a wireless Palm handset owned by me and brought to MIT to be able to show Alexandra Dunn my system.

10. That Appendix D is Alexandra Dunn's Declaration in Support attesting to the above.

11. That my personal computer used for the demonstration at MIT employed the DB2 database which was programmed in part with the C++ code listed in my previous Rule 131 Declaration.

12. That Appendix E hereto documents portions of the C++ code used to program my personal computer used for the demonstration at MIT, categorizing the code as to function.

13. That Appendix F provides maps of the interactive tables used in the demonstration at MIT.

14. That the demonstration at MIT constitutes a second actual reduction to practice of the claimed invention.

15. That the two actual reductions to practice of my claimed invention prior to February 22, 2000 require removal of the Schoenberg and Zak et al. references.

Further deponent sayeth not.

I further declare that all the statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001

of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Date: January 16, 2009

A. Christian Tahan

A. Christian Tahan

APPENDIX A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	A. Christian Tahan	Examiner:	Robert W. Morgan
Application No:	09/784,751	Art Unit:	3626
Filing Date:	02/15/2001		
Title:	Method Of Using a Global Server For Providing Patient Medical Histories To Assist In the Delivery Of Emergency Medical Services		
Atty. Docket:	XWRLD-102		

RULE 132 DECLARATION IN SUPPORT

Commissioner of Patents & Trademarks
U.S. Patent and Trademark Office
P. O. Box 1450
Alexandria, VA 22313-1450

Now comes Samir Tahan and deposes and says:

1. That I understand that A. Christian Tahan has filed a Patent Application entitled METHOD OF USING A GLOBAL SERVER FOR PROVIDING PATIENT MEDICAL HISTORIES TO ASSIST IN THE DELIVERY OF EMERGENCY MEDICAL SERVICES.

2. That I understand that Mr. Tahan has provided a Declaration indicating that he conceived and reduced to practice the claimed invention prior to February 22, 2000.

3. That prior to February 22, 2000, I witnessed an actual reduction to practice of his invention at my house at 1312 Edward Drive, Moncks Corner. S.C. in which a database was populated with patient information, a patient ID was inputted to the system, thereby to give access to the system, that information about the patient was inputted to

the system from a remote site and that the database was queried to provide information back to emergency personnel at the site based on the identity of the patient.

4. That I am an engineer having been schooled in Italy and am conversant with Mr. Tahan's invention and assisted him by providing a server.

5. That at the time I understood wireless communications and the necessity of assisting EMTS with updated information to permit them to assist patients at the site of care.

Further deponent sayeth not.

I further declare that all the statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Date: JAN. 6 - 2009

Samir Tahan

APPENDIX B

rverWindows98.JPG (JPEG Image, 1920x1080 pixels) - Scaled ...

QuestP

etscape2.JPG (JPEG Image, 1920x1080 pixels) - Scaled (30%)

'atientrecord.JPG (JPEG Image, 1920x1080 pixels) - Scaled (30%)

ogin.JPG (JPEG Image, 1920x1080 pixels) - Scaled (30%)

'atientID.JPG (JPEG Image, 1920x1080 pixels) - Scaled (30%)

Databseinfo.JPG (JPEG Image, 1920x1080 pixels) - Scaled (30%)

)atabaseinfo2.JPG (JPEG Image, 1920x1080 pixels) - Scaled (30%)

APPENDIX C

Palm2.JPG (JPEG image, 455x694 pixels) - Scaled (78%)

APPENDIX D

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	A. Christian Tahan	Examiner:	Robert W. Morgan
Application No:	09/784,751	Art Unit:	3626
Filing Date:	02/15/2001		
Title:	Method Of Using a Global Server For Providing Patient Medical Histories To Assist In the Delivery Of Emergency Medical Services		
Atty. Docket:	XWRLD-102		

RULE 132 DECLARATION IN SUPPORT

Commissioner of Patents & Trademarks
U.S. Patent and Trademark Office
P. O. Box 1450
Alexandria, VA 22313-1450

Now comes Alexandra Dunn and deposes and says:

1. That I understand that A. Christian Tahan has filed a Patent Application entitled METHOD OF USING A GLOBAL SERVER FOR PROVIDING PATIENT MEDICAL HISTORIES TO ASSIST IN THE DELIVERY OF EMERGENCY MEDICAL SERVICES.

2. That I understand that Mr. Tahan has provided a Declaration indicating that he conceived and reduced to practice the claimed invention prior to February 22, 2000.

3. That prior to February 22, 2000 I was asked by Chris Tahan to come over to MIT to take a look at his system for providing emergency personnel with information critical to patient treatment, especially at an accident scene.

4. That prior to February 22, 2000 Chris Tahan transported his personal computer and Palm handheld device to a convenient room at MIT where he demonstrated his sytem.

5. That at the time I'd just finished high school and that I was interested in becoming an EMT because my father was a physician who traveled extensively; and I was thus aware of the need for equipment that could provide patient information and treatment options.

6. That prior to February 22, 2000 when I arrived at the MIT classroom I observed Chris Tahan loading patient information into a database on his computer using a specialized form he had provided for data entry.

7. That prior to February 22, 2000 Chris Tahan entered a patient ID number into the computer which accessed database records in which the database records corresponded to patient identification which were displayed on the computer screen.

8. That prior to February 22, 2000 this information was transmitted from his computer to a Palm handheld device, with the patient information downloaded to the device from his computer.

9. That prior to February 22, 2000 the information downloaded to the device was useful in determining patient condition, history and patient treatment.

10. That upon demonstration of Chris Tahan's system it was apparent to me that the system worked to provide useful information to EMT's or others at the scene of an accident to assist in patient treatment.

Further deponent sayeth not.

I further declare that all the statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001

of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Date: 01-14-09

Alexandra Dunn

Alexandra Dunn

APPENDIX E

[Code provided here is for the input of patient information obtained with an identification code at the site of the patient in distress; code below assists in assigning labels to uploaded patient files for storage and retrieval]

```
//*************************************************************
//                                  *
//          FileOfLabels                    *
//                                  *
//*************************************************************
FileOfLabels::FileOfLabels()
        {
        prev_entry=0;
        }
FileOfLabels::~FileOfLabels()
        {
        prev_entry=0;
        }

        if (symb_position[prev_entry].pos_in_file==0UL OR
                        symb_position[0].pos_in_file==0UL)
                merr<<"Unsegmented file. Filtered access not possible";

        if (symb_position[prev_entry].pos_in_file>act_smp AND
                        symb_position[prev_entry].num_sym==sym)
                {
                Assert(prev_entry==0 OR symb_position[prev_entry-
1].pos_in_file<act_smp);
                new_smp_pos=act_smp;
                return (Boolean)TRUE;
                }

        prev_entry++;
        while (prev_entry<symb_position.Dim() AND
                                        symb_position[prev_entry].num_sym!=sym)
                        prev_entry++;

        if(prev_entry==symb_position.Dim())
                {
                new_smp_pos = 0;
                prev_entry = 0;
                return (Boolean)FALSE;
                }
                else    {
                                new_smp_pos=symb_position[prev_entry-1].pos_in_file;
                                return (Boolean)TRUE;
                                }
```

```
        //BINARY search

        t_index step,i;
        i=(symb_position.Dim())/2;
        step=(1+i)/2;

        while(!(symb_position[i].pos_in_file<smp AND
symb_position[i+1].pos_in_file>smp)
                 AND !(symb_position[i].pos_in_file>smp AND i==0))
                {
                if (symb_position[i].pos_in_file<smp)
                        i+=step;
                else i-=step;

                step=(1+step)/2;

                }

        if(i!=0 OR symb_position[i].pos_in_file<smp)
                i++;
        sym=symb_position[i].num_sym;
        i=prev_entry;
        return;
        }


        for(j=0;j<dim;j++)
                {
                Translate_Symbol(tempsymb, symb_position[j].num_sym);
                file<<tempsymb<<" ";
                }
        file<<endl;
        return file;
        }


void GenericFileOfLabels::Reset()
        {
        label= No_Symbol;
        prev_entry=0;
        symb_position.Reset();
        symb_table.Reset();
        return;
        }
```

```
//*************************************************************
//                                    *
//          NTimitLabel                  *
//                                    *
//*************************************************************
Boolean NTimitLabelClass::Initialize(const String &  file_name,
                                     const String &  file_section, const String
&label_ext)
        {
        t_index num_sym,i;

        label_extension=label_ext;

        Assert(label==NTimitLabel);

        const char *list_of_symbols[]=

{"iy","ih","eh","ae","ux","ix","ax","ah","uw","uh","ao","aa","ey","ay","oy","aw",

"ow","l","r","y","w","er","axr","el","em","en","eng","m","n","ng","ch","jh",
                "dh","b","d","dx","nx","g","p","t","k","q","z","zh","v","f","th","s",

"sh","hh","hv","pcl","tcl","kcl","qcl","bcl","dcl","gcl","epi","h#","#h","pau","ax-h"};

        num_sym=63;

        symb_table.Destroy_And_ReDim(num_sym);
        for(i=0;i<num_sym;i++)
                symb_table[i]=list_of_symbols[i];

        return TRUE;
        }


Boolean NTimitLabelClass::Open_Sym(const String & file_name)
        {

        String name,temp;
        ifstream f_lis;
        t_index i=0;
        t_index num_sym=0;
        t_index temp_num;

        prev_entry=0;
        name<<file_name<<"."<<label_extension;
        f_lis.open(name,ios::in|ios::nocreate);
```

```
        if(f_lis.fail())

        while (NOT f_lis.eof())
                {
                f_lis>>temp_num;
                f_lis>>temp_num;
                f_lis>>temp;
                if(!(f_lis.eof() AND temp[0]==EOF)) num_sym++;
                }


        f_lis.clear();
        f_lis.seekg(0,ios::beg);

        if (num_sym==0)
                merr<<"Empty file of ID transcription "<<name;

        symb_position.Destroy_And_ReDim(num_sym);
        for (i=0;i<num_sym;i++)
                {
                f_lis>>temp_num;
                f_lis>>symb_position[i].pos_in_file;
                f_lis>>temp;
                symb_position[i].num_sym=Translate_Symbol(temp);
                }

        f_lis.close();

        return TRUE;
        }


//*************************************************************
//                                      *
//              NTimitReducedLabel                      *
//                                      *
//*************************************************************

Boolean NTimitReducedLabelClass::Open_Sym(const String & file_name)
        {

        String name,temp;
        ifstream f_lis;
        t_index i=0;
        t_index num_sym=0;
        t_index temp_num;
```

```
        prev_entry=0;
        name<<file_name<<"."<<label_extension;
        f_lis.open(name,ios::in|ios::nocreate);
        if(f_lis.fail())
                merr<<"Could not open ID transcription file: "<<name;


        while (NOT f_lis.eof())
                {
                f_lis>>temp_num;
                f_lis>>temp_num;
                f_lis>>temp;
                if(!(f_lis.eof() AND temp[0]==EOF)) num_sym++;
                }


        f_lis.clear();
        f_lis.seekg(0,ios::beg);

        if (num_sym==0)
                merr<<"Empty file of ID transcription "<<name;

        symb_position.Destroy_And_ReDim(num_sym);
        for (i=0;i<num_sym;i++)
                {
                f_lis>>temp_num;
                f_lis>>symb_position[i].pos_in_file;
                f_lis>>temp;
                symb_position[i].num_sym=Translate_Symbol(temp);
                }

        f_lis.close();

        return TRUE;
        }

t_index NTimitReducedLabelClass::Translate_Symbol(const String & sym) const
        {
        t_index num=0;
        t_index len_sym;

        Assert(label == NTimitReducedLabel OR label ==AtisLabel);

        len_sym=symb_table.Dim();
        while(num<len_sym AND symb_table[num]!=sym)
```

```
            num++;

        if(num==len_sym)
            {
            if(sym=="ux") num=7;   else
            if(sym=="el") num=16;  else
            if(sym=="axr") num=20; else
            if(sym=="ax-h") num=5; else
            if(sym=="em") num=21;  else
            if(sym=="en") num=22;  else
            if(sym=="nx") num=22;  else
            if(sym=="eng") num=23; else
            if(sym=="q") num=32;   else
            if(sym=="hv") num=40;  else
            if(sym=="pcl") num=41; else
            if(sym=="tcl") num=41; else
            if(sym=="kcl") num=41; else
            if(sym=="qcl") num=41; else
            if(sym=="bcl") num=42; else
            if(sym=="dcl") num=42; else
            if(sym=="gcl") num=42; else
            if(sym=="***") num=100; else  //separator

            if(sym=="#h" OR sym=="h#" OR sym == "pau" )
                    num=44;
            else {
                    merr<<"unknown symbol of NTIMIT. Symbol: "<<sym;
                    }
            }


        return num;
        }


Boolean NTimitReducedLabelClass::Initialize(const String & file_name,
                                                        const String &
section_name, const String &label_ext)
        {
        t_index num_sym,i;

        label_extension=label_ext;

        Assert(label==NTimitReducedLabel);

        const char *list_of_symbols[]=
```

```
        {"iy","ih","eh","ae","ix","ax","ah","uw","uh","ao","aa","ey","ay","oy","aw","ow",
                "l","r","y","w","er","m","n","ng","ch","jh","dh","b","d","g","p","t",
                "k","z","zh","v","f","th","s","sh","hh","cl","vcl","epi","sil","dx"};

        num_sym=46;

        symb_table.Destroy_And_ReDim(num_sym);
        for(i=0;i<num_sym;i++)
                symb_table[i]=list_of_symbols[i];

        return TRUE;
        }

//*************************************************************
//                                      *
//              NTimit39Label                   *
//                                      *
//*************************************************************
Boolean NTimit39LabelClass::Open_Sym(const String & file_name)
        {

        String name,temp;
        ifstream f_lis;
        t_index i=0;
        t_index num_sym=0;
        t_index temp_num;

        prev_entry=0;
        name<<file_name<<"."<<label_extension;
        f_lis.open(name,ios::in|ios::nocreate);
        if(f_lis.fail())
                merr<<"Could not open ID transcription file: "<<name;


        while (NOT f_lis.eof())
                {
                f_lis>>temp_num;
                f_lis>>temp_num;
                f_lis>>temp;
                if(!(f_lis.eof() AND temp[0]==EOF)) num_sym++;
                }


        f_lis.clear();
        f_lis.seekg(0,ios::beg);
```

```
        if (num_sym==0)
                merr<<"Empty file of ID transcription "<<name;

        symb_position.Destroy_And_ReDim(num_sym);
        for (i=0;i<num_sym;i++)
                {
                f_lis>>temp_num;
                f_lis>>symb_position[i].pos_in_file;
                f_lis>>temp;
                symb_position[i].num_sym=Translate_Symbol(temp);
                }

        f_lis.close();

        return TRUE;
        }

t_index NTimit39LabelClass::Translate_Symbol(const String & sym) const
        {
        t_index num=0;
        t_index len_sym;

        Assert(label == NTimit39Label);

        len_sym=symb_table.Dim();
        while(num<len_sym AND symb_table[num]!=sym)
                num++;

        if(num==len_sym)
                {
                if(sym=="#h" OR sym=="h#" OR sym == "pau" OR sym == "pcl"
                        OR sym == "tcl" OR sym == "kcl" OR sym == "bcl" OR sym ==
"dcl"
                        OR sym == "gcl" OR sym == "qcl" OR sym == "epi") num=37;
//sil

                else if(sym=="ux") num=5;               //uw
                else if(sym=="el") num=13;              // l
                else if(sym=="axr") num=17;             //er
                else if(sym=="ax-h" OR sym=="ax") num=4; //ah
                else if(sym=="em") num=18;              // m
                else if(sym=="en" OR sym=="nx") num=19;  // n
                else if(sym=="eng") num=20;             //ng
                else if(sym=="q") num=29;                               // k
                else if(sym=="hv") num=36;                              //hh
```

```
        else if(sym=="ao") num=7;                    //aa
        else if(sym=="ix") num=1;                    //ih
        else if(sym=="zh") num=35;                   //sh
        else if(sym=="***") num=100;                 // separator

        else {
                merr<<"unknown symbol of NTIMIT. Symbol: "<<sym;
                }
}

return num;
}


Boolean NTimit39LabelClass::Initialize(const String & file_name,
                                                        const String &
section_name, const String &label_ext)
        {
        t_index num_sym,i;

        label_extension=label_ext;

        Assert(label==NTimit39Label);

        const char *list_of_symbols[]=
                {"iy","ih","eh","ae","ah","uw","uh","aa","ey","ay","oy","aw","ow",
                "l","r","y","w","er","m","n","ng","ch","jh","dh","b","d","g",
                "p","t","k","z","v","f","th","s","sh","hh","sil","dx"};

        num_sym=39;

        symb_table.Destroy_And_ReDim(num_sym);
        for(i=0;i<num_sym;i++)
                symb_table[i]=list_of_symbols[i];

        return TRUE;
        }


Boolean AtisReducedLabelsClass::Initialize(const String &  file_name,
                                                        const String &
section_name, const String &label_ext)
        {
        t_index num_sym,i;
```

```
        label_extension=label_ext;

        Assert(label== AtisReducedLabel);

        const char *list_of_symbols[]=
                {"iy","ih","eh","ae","ah","uw","uh","aa","ey","ay","oy","aw","ow",
                 "l","r","y","w","m","n","ch","jh","dh","b","d","g",
                 "p","t","k","z","v","f","th","s","sh","hh","sil"};

        num_sym=36;


        symb_table.Destroy_And_ReDim(num_sym);

        for(i=0;i<num_sym;i++)
                symb_table[i]=list_of_symbols[i];

        return TRUE;
        }

Boolean AtisReducedLabelsClass::Open_Sym(const String &file_name)
{
        String name,temp;
        ifstream f_lis;
        t_index i=0;
        t_index num_sym=0;
        t_index temp_num;

        prev_entry=0;
        name<<file_name<<"."<<label_extension;
        f_lis.open(name,ios::in|ios::nocreate);
        if(f_lis.fail())
                merr<<"Could not open ID transcription file: "<<name;


        while (NOT f_lis.eof())
                {
                f_lis>>temp_num;
                f_lis>>temp_num;
                f_lis>>temp;
                if(!(f_lis.eof() AND temp[0]==EOF)) num_sym++;
                }


        f_lis.clear();
```

```
        f_lis.seekg(0,ios::beg);

        if (num_sym==0)
                merr<<"Empty file of ID transcription "<<name;

        symb_position.Destroy_And_ReDim(num_sym);
        for (i=0;i<num_sym;i++)
                {
                f_lis>>temp_num;
                f_lis>>symb_position[i].pos_in_file;
                f_lis>>temp;
                symb_position[i].num_sym=Translate_Symbol(temp);
                }


        f_lis.close();

        return TRUE;
        }

t_index AtisReducedLabelsClass::Translate_Symbol(const String & sym) const
        {
        t_index num=0;
        t_index len_sym;

        Assert(label ==AtisReducedLabel);

        len_sym=symb_table.Dim();
        while(num<len_sym AND symb_table[num]!=sym)
                num++;

        if(num==len_sym)
                {
                        if(sym=="ao") num=7;    else // aa
                        if(sym=="ix") num=1;    else // ih
                        if(sym=="nx") num=18;   else // n
                        if(sym=="ax") num=4;    else // ah
                        if(sym=="zh") num=33;   else // sh
                        if(sym=="***") num=100;  else // separator

                                {
                                merr<<"unknown symbol of ATISLabel. Symbol: "<<sym;
                                }
                }

        return num;
```

```
}


Boolean AtisLabelsClass::Open_Sym(const String & file_name)
{

        String name,temp;
        ifstream f_lis;
        t_index i=0;
        t_index num_sym=0;

        prev_entry=0;
        name<<file_name<<"."<<label_extension;
        f_lis.open(name,ios::in|ios::nocreate);
        if(f_lis.fail())
                merr<<"Could not open ID transcription file: "<<name;

        while (!f_lis.eof())
                {
                f_lis>>temp;
                if(!(f_lis.eof() AND temp[0]==EOF)) num_sym++;
                }
        f_lis.clear();
        f_lis.seekg(0,ios::beg);

        if (num_sym==0)
                merr<<"Empty file of ID transcription "<<name;

        symb_position.Destroy_And_ReDim(num_sym);
        for (i=0;i<num_sym;i++)
                {
                f_lis>>temp;
                symb_position[i].num_sym=Translate_Symbol(temp);
                }

        f_lis.close();

        return TRUE;
        }


Boolean AtisLabelsClass::Initialize(const String &  file_name,
                                                        const String &
section_name, const String &label_ext)
```

```
{
t_index num_sym,i;

label_extension=label_ext;

Assert(label== AtisLabel);
{
const char *list_of_symbols[]=

{"iy","ih","ix","eh","ae","ao","ax","uw","uh","aa","ey","ay","oy","aw","ow",
        "l","r","y","w","er","m","n","ng","nx","ch","jh","dh","b","d","g",
        "p","t","k","z","v","f","th","s","sh","zh","hh","sil"};


num_sym=42;

symb_table.Destroy_And_ReDim(num_sym);

for(i=0;i<num_sym;i++)
symb_table[i]=list_of_symbols[i];

}

return TRUE;
}

Boolean ApasciLabelsClass::Open_Sym(const String & file_name)
{
        String name,temp;
        ifstream f_lis;
        t_index i=0;
        t_index num_sym=0;
        t_index temp_num;

        prev_entry=0;
        name<<file_name<<"."<<label_extension;
        f_lis.open(name,ios::in|ios::nocreate);
        if(f_lis.fail())
                merr<<"Could not open ID transcription file: "<<name;


        while (NOT f_lis.eof())
                {
                f_lis>>temp_num;
                f_lis>>temp_num;
                f_lis>>temp;
```

```
        if(!(f_lis.eof() AND temp[0]==EOF)) num_sym++;
        }


    f_lis.clear();
    f_lis.seekg(0,ios::beg);

    if (num_sym==0)
        merr<<"Empty file of ID transcription "<<name;

    symb_position.Destroy_And_ReDim(num_sym);
    for (i=0;i<num_sym;i++)
        {
        f_lis>>temp_num;
        f_lis>>symb_position[i].pos_in_file;
        f_lis>>temp;
        symb_position[i].num_sym=Translate_Symbol(temp);
        }

    f_lis.close();

    return TRUE;
    }

t_index ApasciLabelsClass::Translate_Symbol(const String & sym) const
    {
    t_index num=0;
    t_index len_sym;

    Assert(label == ApasciLabel);

    len_sym=symb_table.Dim();
    while(num<len_sym AND symb_table[num]!=sym)
        num++;

    if(num==len_sym)
        {
        if(sym=="E") num=1;                    // e
        else if(sym=="O") num=3;               // o
        else if(sym=="@bg") num=48;                              // sil
        else if(sym=="***") num=100;                          // separator
        else {
            merr<<"unknown symbol of APASCI Symbol: "<<sym;
            }
    }
```

```
        return num;
        }


Boolean ApasciLabelsClass::Initialize(const String &  file_name,
                                                        const String &
section_name, const String &label_ext)
        {
        t_index num_sym,i;

        label_extension=label_ext;

        Assert(label== ApasciLabel);

        const char *list_of_symbols[]=
                {"a","e","i","o","u","f","v","s","z","S","ff","vv","ss",
                 "SS","tS","dZ","ts","dz","ttS","ddZ","tts","ddz","j","w","p","t","k","b",
                 "d","g","pp","tt","kk","bb","dd","gg","m","n","J","mm","nn","JJ","l","r",
                 "L","ll","rr","LL","sil", "@sch"};


        num_sym=50;

        symb_table.Destroy_And_ReDim(num_sym);

        for(i=0;i<num_sym;i++)
        symb_table[i]=list_of_symbols[i];

        return TRUE;
        }


Boolean ApasciReducedLabelsClass::Open_Sym(const String & file_name)
        {

        String name,temp;
        ifstream f_lis;
        t_index i=0;
        t_index num_sym=0;
        t_index temp_num;

        prev_entry=0;
        name<<file_name<<"."<<label_extension;
        f_lis.open(name,ios::in|ios::nocreate);
        if(f_lis.fail())
                merr<<"Could not open ID transcription file: "<<name;
```

```
        while (NOT f_lis.eof())
                {
                f_lis>>temp_num;
                f_lis>>temp_num;
                f_lis>>temp;
                if(!(f_lis.eof() AND temp[0]==EOF)) num_sym++;
                }


        f_lis.clear();
        f_lis.seekg(0,ios::beg);

        if (num_sym==0)
                merr<<"Empty file of ID transcription "<<name;

        symb_position.Destroy_And_ReDim(num_sym);
        for (i=0;i<num_sym;i++)
                {
                f_lis>>temp_num;
                f_lis>>symb_position[i].pos_in_file;
                f_lis>>temp;
                symb_position[i].num_sym=Translate_Symbol(temp);
                }

        f_lis.close();

        return TRUE;
        }

t_index ApasciReducedLabelsClass::Translate_Symbol(const String & sym) const
        {
        t_index num=0;
        t_index len_sym;

        Assert(label == ApasciReducedLabel);

        len_sym=symb_table.Dim();
        while(num<len_sym AND symb_table[num]!=sym)
                num++;

        if(num==len_sym)
                {
                if(sym=="E") num=1;                     // e
                else if(sym=="O") num=3;                // o
```

```
        else if(sym=="ff") num=5;              // f
        else if(sym=="vv") num=6;                       // v
        else if(sym=="ss") num=7;              // s
        else if(sym=="SS") num=9;                       // S
        else if(sym=="ttS") num=10;            //tS
        else if(sym=="ddZ") num=11;                             //dZ
        else if(sym=="tts") num=12;                     //ts
        else if(sym=="ddz") num=13;                             //dz
        else if(sym=="pp") num=16;                      // p
        else if(sym=="tt") num=17;                      // t
        else if(sym=="kk") num=18;              // k
        else if(sym=="bb") num=19;              // b
        else if(sym=="dd") num=20;              // d
        else if(sym=="gg") num=21;              // g
        else if(sym=="mm") num=22;                      // m
        else if(sym=="nn") num=23;              // n
        else if(sym=="JJ") num=24;              // J
        else if(sym=="ll") num=25;              // l
        else if(sym=="rr") num=26;              // r
        else if(sym=="LL") num=27;              // L
        else if(sym=="@bg") num=28;                     // sil
        else if(sym=="***") num=100;                    // separator
        else {
                merr<<"unknown symbol of APASCI Symbol: "<<sym;
                }
}

return num;
}


Boolean ApasciReducedLabelsClass::Initialize(const String & file_name,
                                                const String &
section_name, const String &label_ext)
{
t_index num_sym,i;

label_extension=label_ext;

Assert(label==ApasciReducedLabel);

const char *list_of_symbols[]=
        {"a","e","i","o","u","f","v","s","z","S","tS","dZ","ts","dz","j","w","p",
         "t","k","b","d","g","m","n","J","[","r","L","sil","@sch"};

num_sym=30;
```

```
        symb_table.Destroy_And_ReDim(num_sym);
        for(i=0;i<num_sym;i++)
                symb_table[i]=list_of_symbols[i];

        return TRUE;
        }


//*************************************************************
//                                    *
//            LabelTrans                *
//                                    *
//*************************************************************

Boolean CustomLabelsFromFile::Initialize(const String &  file_name,
                                                                const
String &  section_name, const String &label_ext)
        {
        merr<<"this function must be implemented";
        return TRUE;
        }

Boolean CustomLabelsFromFile::Open_Sym(const String & file_name)
        {
        merr<<"this function must be implemented";
        return TRUE;
        }
```

[Code here is related to the uploading of patient information from the remote site to the database; code below tries to facilitate storage and finding space for a large file; part II is code that tries to update files, replacing old files with uploaded new files]

```
//*************************************************************
// Create Storage Space - excluding actual data arrays

        // storage for min and max q values
        qHi.Destroy_And_ReDim(T);
        qLo.Destroy_And_ReDim(T);

        // dimensionate beta and obs_lprob
        beta.Destroy_And_ReDim(Q);
        obs_lprob.Destroy_And_ReDim(Q);
        for(q=0;q<Q;q++)
                {
                beta[q].Destroy_And_ReDim(T);
                obs_lprob[q].Destroy_And_ReDim(T);
                }
        maxP.Destroy_And_ReDim(Q);                  // for calculating beam width

        act_HMM = &HMM_defs[label_idxes_list[Q-1]];
        Nq = act_HMM->num_states;
        beta[Q-1][T-1].Destroy_And_ReDim(Nq);
        beta[Q-1][T-1][Nq-1]=0.0;

        for (i=1;i<Nq-1;i++)
                beta[Q-1][T-1][i] = act_HMM->trans_mat[i][Nq-1];

        beta[Q-1][T-1][0]=LOGZERO;
        qHi[T-1] = qLo[T-1] = Q-1;
        Compute_Obs_LProbs(whole_file[T-1], T-1, qHi[T-1],
                                                                qLo[T-
1], label_idxes_list);
        Assert(T>=2);
        for (t=T-2;t!=(t_index)(-1);t--)
                {
                gMax = LOGZERO;    // max value of beta at time t
                if(t>=qHi[t+1]) startq=qHi[t+1];
                else startq =  t;

                if (0==qLo[t+1]) endq = 0;
                else  endq = qLo[t+1]-1;

                Assert(startq>=endq);
```

```
for (q=startq;q!=(t_index)(endq-1);q--)
        {
        lMax = LOGZERO; // max value of beta in model q
        act_HMM = &HMM_defs[label_idxes_list[q]];
        Nq = act_HMM->num_states;
        // create vec for beta vals
        beta[q][t].Destroy_And_ReDim(Nq);
        outprob = obs_lprob[q][t+1];

        if (q==startq) beta[q][t][Nq-1]= LOGZERO;
        else beta[q][t][Nq-1]= beta[q+1][t][0];

        Assert(Nq>=2);
        for (i=Nq-2;i!=(t_index)(-1);i--)
                {
x = act_HMM->trans_mat[i][Nq-1] + beta[q][t][Nq-1];
                if (q>=qLo[t+1] AND q<=qHi[t+1])
                        for (j=1;j<Nq-1;j++)
                                {
    a = act_HMM->trans_mat[i][j];
                                y = beta[q][t+1][j];
                                if (a>LOGSMALL AND y>LOGSMALL)
                                        x = LogAdd(x,a+outprob[j]+y);
                                } // endfor j

                beta[q][t][i] = x;
                if (x>lMax) lMax = x;
                if (x>gMax)
                        {
                        gMax = x;
                        q_at_gMax = q;
                        }
                } // endfor i
        maxP[q] = lMax;
        } // endfor q

last_q = endq;

while (gMax-maxP[startq] > pruning_threshold)
        startq-=1;    // lower startq till threshold reached
qHi[t] = startq;
while ( ((gMax-maxP[endq]) > pruning_threshold) AND endq<t)
        endq+=1;      // raise endq till thresh reached
qLo[t] = endq;

Compute_Obs_LProbs(whole_file[t], t, qHi[t], qLo[t], label_idxes_list);
```

```
            } // endfor t

        // compute total probability pr
        pr = LOGZERO;
        outprob = obs_lprob[0][0];
        for (j=1;j<Nq-1;j++)
            {
            a = act_HMM->trans_mat[0][j];
            y = beta[last_q][0][j];
            if ( (a>LOGSMALL) AND (y>LOGSMALL) )
                pr = LogAdd(pr,a+outprob[j]+y);
            }

        if (LOGZERO >= pr)
            {
            mwarn<<"Prune threshold = "<<pruning_threshold<<" too small.";
            return pr;
            }

        return pr;
        }

// Setotprob: allocate and calculate otprob matrix at time t
void ModelsSimultaneousTraining::Compute_Obs_LProbs(const VetDouble& obs,
                                const t_index t, const t_index beam_top,
                                const t_index beam_bottom, const VetULong&
label_idxes_list)
        {
        t_signed q;
        t_index j, Nq, endq;
        VetDouble temp_dvet;
        EmbCodebook *act_HMM;

        if (0==beam_bottom) endq = 0;
        else endq = beam_bottom-1;

        for (q=beam_top; q>=(t_signed)endq; q--)
            {
            act_HMM = &HMM_defs[label_idxes_list[q]];
            Nq = act_HMM->num_states;

            obs_lprob[q][t].Destroy_And_ReDim(Nq-1);
            for (j=1;j<Nq-1;j++)
                obs_lprob[q][t][j]=(*act_HMM)[j-1].Obs_LProb(obs);
            }
```

```
        return ;}
PartII
/ ******************    UPDATE MODELS    ********************


void ModelsSimultaneousTraining::Store_Statistic_Accs(const String& accs_file)
        {
        t_index i,j,h,k,z,Nh,Mh;
        t_index obs_size;
   ofstream file;

        file.open(accs_file);
        file.precision(OUTPUT_SIZE);

        for(h=0;h<HMM_accs.Dim();h++)
                {
                Nh=HMM_accs[h].num_states;
                Mh=HMM_accs[h].num_mixes;

                file<<"file: "<<h<<"\n\n";
          file<<"num_istances= "<<HMM_accs[h].num_istances<<"\n";
                file<<"num_states= "<<Nh<<"\n";
                file<<"num_mixes= "<<Mh<<"\n\n";
                file<<"tran:\n";

                for(i=0;i<Nh-1;i++)
                        {
                        for(j=1;j<Nh;j++)
                      file<<HMM_accs[h].tran[i][j]<<" ";
                        file<<"\n";
                        }

                file<<"\nocc: ";
                for(i=0;i<Nh-1;i++)
                        file<<HMM_accs[h].occ[i]<<" ";

          obs_size=HMM_accs[h].mu[0][0].Dim();

                file<<"\n\nmu:\n";
                for(i=1;i<Nh-1;i++)
                        for(j=0;j<Mh;j++)
                                {
                                for(k=0;k<obs_size;k++)
                                        file<<HMM_accs[h].mu[i][j][k]<<" ";
                                file<<"\n";
```

```
                    }

            if(HMM_accs[h].full_cov.Dim()!=0)
                    {
                    file<<"\nfull_cov: \n";
                    for(i=1;i<Nh-1;i++)
                       for(j=0;j<Mh;j++)
                    for(k=0;k<obs_size;k++)
                                        {
                                        for(z=k;z<obs_size;z++)

        file<<HMM_accs[h].full_cov[i][j][k][z]<<" ";
                                        file<<"\n";
                                        }
                    }
                else{
                        file<<"\n\ndiag_va: \n";
                   for(i=1;i<Nh-1;i++)
                      for(j=0;j<Mh;j++)
                   {
                                    for(k=0;k<obs_size;k++)
                                        file<<HMM_accs[h].diag_va[i][j][k]<<" ";
                   file<<"\n";
                   }
            }

                file<<"\nc: \n";
                for(i=1;i<Nh-1;i++)
                        {
                        for(j=0;j<Mh;j++)
                                file<<HMM_accs[h].c[i][j]<<" ";
                        file<<"\n";
                        }
                file<<"\n";
                }

        file.close();
        return;
        }

void ModelsSimultaneousTraining::Load_Models_Parameters()
        {
        t_index symbol, num_symbols;
        t_index vec_size;
        String buffer;
        Boolean use_full_cov;
```

```
        ifstream init_spcf;

        init_spcf.open(models_file_input, ios::in|ios::nocreate);
        Read_Data_File_Header (init_spcf, vec_size, use_full_cov);

        if(features.Feature_Vet_Dim()!=vec_size)
                merr<<"Not compatible statistics dimension with initialized acoustic
models";

        Write_Header_Of_File_Model(models_file_output, dbase.Snd_Type(),
                dbase.Label_Type(), dbase.Db_File_List_Name(),
dbase.Window_Lenght(),
                dbase.Window_Overlap(), vec_size, use_full_cov);

        num_symbols = HMM_defs.Dim();
        HMM_accs.Destroy_And_ReDim(num_symbols);

        for(symbol=0; symbol<num_symbols; symbol++)
                {
                HMM_defs[symbol].file=symbol;
                HMM_defs[symbol].stat_dim=vec_size;
                HMM_defs[symbol].full_covariance=use_full_cov;
                HMM_defs[symbol].Read(init_spcf, use_full_cov);

        HMM_accs[symbol].Configure(HMM_defs[symbol].num_states,HMM_defs[sym
bol].num_gauss,
                                vec_size, use_full_cov);
                }

        return;

        }


void ModelsSimultaneousTraining::Load_Statistic_Accs(const String& accs_file)
        {
        t_index i,j,h,k,z,Nh,Mh;
        t_index obs_size, file;
        ifstream file;
   String buffer;
        t_real val;

        file.open(accs_file,ios::in|ios::nocreate);
        if(file.fail())
                merr<<"Could not open file of statistics accumulators.";
```

```
file.precision(OUTPUT_SIZE);

for(h=0;h<HMM_accs.Dim();h++)
        {
        file>>buffer;
        file>>file;
        Assert(file==h);

        file>>buffer;
        file>>HMM_accs[h].num_istances;
        file>>buffer;
        file>>Nh;
        file>>buffer;
        file>>Mh;
        file>>buffer;
        for(i=0;i<Nh-1;i++)
                for(j=1;j<Nh;j++)
                        {
                        file>>val;
                        HMM_accs[h].tran[i][j]+=val;
                        }

        file>>buffer;
        for(i=0;i<Nh-1;i++)
                {
                file>>val;
                HMM_accs[h].occ[i]+=val;
                }

        obs_size=HMM_accs[h].mu[0][0].Dim();
  file>>buffer;
        for(i=1;i<Nh-1;i++)
                for(j=0;j<Mh;j++)
                        for(k=0;k<obs_size;k++)
                                {
                                file>>val;
                                HMM_accs[h].mu[i][j][k]+=val;
                                }

        file>>buffer;
        if(buffer=="full_cov:")
                for(i=1;i<Nh-1;i++)
                        for(j=0;j<Mh;j++)
                                for(k=0;k<obs_size;k++)
                                        for(z=k;z<obs_size;z++)
                                                {
```

```
                                                    file>>val;

HMM_accs[h].full_cov[i][j][k][z]+=val;
                                                    }
        else for(i=1;i<Nh-1;i++)
                        for(j=0;j<Mh;j++)
                    for(k=0;k<obs_size;k++)
                                    {
                                    file>>val;
                                HMM_accs[h].diag_va[i][j][k]+=val;
                                    }

        file>>buffer;
        for(i=1;i<Nh-1;i++)
                for(j=0;j<Mh;j++)
    {
                file>>val;
                HMM_accs[h].c[i][j]+=val;
                }
        }

file.close();
return;
}
```

[Code here is related to the retrieval of the patient's file from the database for use by emergency personnel, hospital personnel, or insurers; since the wrong files of a particular patient could be downloaded, a waste of time, since the files were similar in appearance and stored in near-by locations with the other code submitted, the code below assists in saving the data as templates (a bundling format for similar data). The code was to allow for the assigning of specific parameters to different data sets to limit retrieval errors. Specific header files are used to create the templates.]

```
//*************************************************************
// Utilizing templates

#ifndef _HYPOLIST_HPP_
#define _HYPOLIST_HPP_

template<class T>
Boolean SparseList<T>::Has_No_Kids(t_ptr node) const
        {
        //dummy definition so that at least one instance of
        //ImpObjectList<T> and linker can find the member function of the class
        ImpObjectList<T> dummy;

        return((Boolean)(List[node].num_kids==0));
        }


template<class T>
SparseList<T>::SparseList(natural chunk)
        {
        chunk_size=chunk;
        free_list=0;
        start_list=0;
        dim_free=0;

        //allocate the nihil=0 element this can't be used
        List.Destroy_And_ReDim(1);

        return;
        }


template<class T>
void SparseList<T>::Restart()
        {
   free_list=0;
        start_list=0;
```

```
        dim_free=0;

        //allocate the nihil=0 element this can't be used
        List.Destroy_And_ReDim(1);
        return;
        }


template<class T>
void SparseList<T>::Reset()
        {
        free_list=0;
        start_list=0;
        dim_free=0;

        return;
        }


template<class T>
void SparseList<T>::Allocate_Mem()
        {
        natural i;
        t_ptr temp=List.Dim();

        List.Save_And_ReDim(chunk_size+temp);

        //link the node of free list
        for (i=temp; i<temp+chunk_size-1; i++)
                List[i].link=i+1;

        List[List.Dim()-1].link=free_list;
        free_list=temp;
        dim_free+=chunk_size;

        return;
        }


template<class T>
t_ptr SparseList<T>::Create(const T & info,t_ptr parent)
        {
   t_ptr temp;

        if (free_list==0)
                Allocate_Mem();
```

```
        //get one node from free list
        temp=free_list;
        free_list=List[temp].link;
        dim_free--;

        //verify that free_node is really free
        Assert(List[temp].num_kids==Node::Kids_Of_Free_Node());
        List[temp].link=parent;
        List[temp].num_kids=0;
        List[temp].info=info;
        List[parent].num_kids++;

        return temp;
        }


template<class T>
t_index SparseList<T>::Num_Node() const
        {
        return (List.Dim()-dim_free-1);
        }


template<class T>
t_ptr SparseList<T>::Next(t_ptr son) const
        {
        Assert (son>0);
        return List[son].link;
        }


template<class T>
void SparseList<T>::Destroy_Node(t_ptr node)
        {
  Assert(node>0);

        if (List[node].num_kids>0)
                merr<<"Attempt to destroy referenced node";

        //decrease parent's num_kids
        List[Next(node)].num_kids--;
        //add to free list
        List[node].link=free_list;
        free_list=node;
        List[free_list].num_kids=Node::Kids_Of_Free_Node();
```

```
        dim_free++;

        return;
        }


template<class T>
void SparseList<T>::Backtrack_From(ImpObjectList<T> & sequence, t_ptr node)
        {
        t_index i=0;

        Assert(node>0);
        sequence.Reset();
        do      {
                sequence.Save_And_ReDim(i+1);
                sequence[i]=(*this)[node];
                i++;
                node=Next(node);
                }
        while ( node!=0);

        // eliminate phantom node
        // is the following instruction necessary in order to eliminate
        // silence and duplicated typo?
        sequence.Save_And_ReDim(i-1);
        T temp;
        for (i=0; i<sequence.Dim()/2; i++)
                {
                temp=sequence[i];
                sequence[i]=sequence[sequence.Dim()-i-1];
                sequence[sequence.Dim()-i-1]=temp;
                }

        return;
        }


template<class T>
void SparseList<T>::Destroy_Branch(t_ptr node)
        {
        t_ptr temp;
        Assert(node>0);

        if (List[node].num_kids>0)
                merr<<"Attempt to destroy referenced node";
```

```
        do      {
                temp=Next(node);
                Destroy_Node(node);
                node=temp;
                }
        while (List[node].num_kids==0 AND node!=0);

        return;
        }


template<class T>
T & SparseList<T>::operator[](const t_ptr son)
        {
        Assert(son>0);
        Assert(List[son].num_kids != Node::Kids_Of_Free_Node());

        return List[son].info;
        }


template<class T>
const T & SparseList<T>::operator[](const t_ptr son)const
        {
        Assert(son>0);
        Assert(List[son].num_kids != Node::Kids_Of_Free_Node());

        return List[son].info;
        }


template<class T>
WellTree<T>::~WellTree()
        {
        l_list.Reset();
        leaves_dir.Reset();
        kid_dir.Reset();
        }

template<class T>
void WellTree<T>::Reset()
        {
        leaves_dir.Reset();
        kid_dir.Reset();
        l_list.Restart() ;
        }
```

```
//needed by Viterbi.num_hypotesis
template<class T>
inline t_index WellTree<T>::Num_Elements() const
        {
   return (l_list.Num_Node());
        }


template<class T>
inline t_index WellTree<T>::Kids_Dim() const
        {
        return kid_dir.Dim();
        }


template<class T>
inline t_index WellTree<T>::Leaves_Dim() const
        {
        return leaves_dir.Dim();
        }


template<class T>
void WellTree<T>::ReDim_Leaves_Dir_To(const t_index ix)
        {
        leaves_dir.Save_And_ReDim(ix);

        return;
        }


template<class T>
inline void WellTree<T>::Exchange_Leaves_Indexes(const t_index i,
                                                 const t_index j)
        {
  t_index aux;
        aux=leaves_dir[i];
        leaves_dir[i]=leaves_dir[j];
        leaves_dir[j]=aux;

        return;
        }
```

```
template<class T>
Boolean WellTree<T>::Check_Kid_Presence_And_Get_Num(const T & act_kid,
                                                    p_kid & kid_idx)
        {
        t_index i=0;
        t_index kid_num;

        kid_num = kid_dir.Dim();
        if (kid_num==0)
                return (Boolean) FALSE;
        else{
                while (i<kid_num AND act_kid!=l_list[kid_dir[i]])
                        i++;

                if (i==kid_num)
                        return (Boolean)FALSE;
                else{
                        kid_idx=i;
                        return (Boolean)TRUE;
                        }
                } // end of else

        }


template<class T>
inline const T& WellTree<T>::Get_Leaf_Info(const p_leaf leaf)const
        {
        return (l_list[leaves_dir[leaf]]);
        }

template<class T>
inline T& WellTree<T>::Get_Leaf_Info(const p_leaf leaf)
        {
        return (l_list[leaves_dir[leaf]]);
        }

template<class T>
inline const T& WellTree<T>::Get_Kid_Info(const p_kid kid)const
        {
        return (l_list[kid_dir[kid]]);
        }

template<class T>
inline T& WellTree<T>::Get_Kid_Info(const p_kid kid)
        {
```

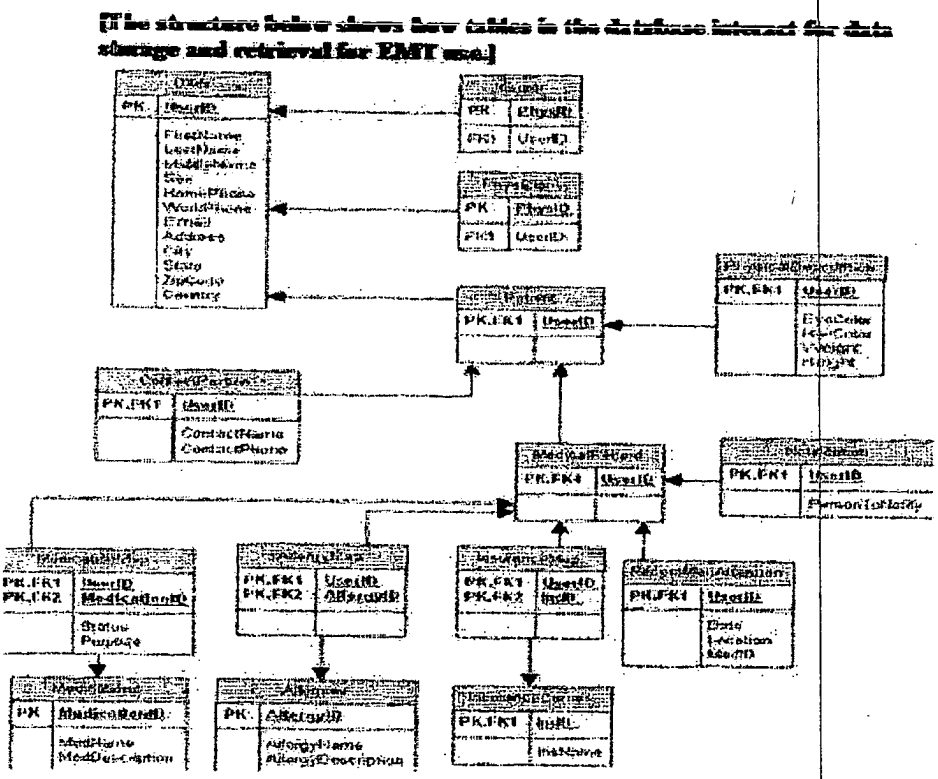
```
        return (l_list[kid_dir[kid]]);
        }

template<class T>
inline void WellTree<T>::Create_First_Leaf_Of_Tree(const T& info)
        {
        //if no elements in tree create leaf
        Assert(l_list.Num_Node()==0);
        leaves_dir.Destroy_And_ReDim(1);
        //0 pointer is NULL
        leaves_dir[0]=l_list.Create(info,0);

        return;
        }


template<class T>
inline void WellTree<T>::Add_Kid_To_Leaf(const T& info, p_leaf leaf)
        {
        //if no elements in tree create a kid
        if (l_list.Num_Node()==0)
                {
                kid_dir.Destroy_And_ReDim(1);
                //0 pointer is NULL
                kid_dir[0]=l_list.Create(info,0);
                return;
                }

        //abort if tree not empty and no leaves
        Assert(l_list.Num_Node()>0 AND leaves_dir.Dim()!=0);

        //abort if more than one well created
        Assert(l_list.Num_Node()>0 AND leaves_dir[leaf]!=0);

        t_index kid_dim=kid_dir.Dim();
        kid_dir.Save_And_ReDim(kid_dim+1);

        kid_dir[kid_dim]=l_list.Create(info,leaves_dir[leaf]);
        return;
        }


template<class T>
void WellTree<T>::Prune_All_Dead_Leaf()
        {
  t_index i;
```

```
        t_index num_leaves=leaves_dir.Dim();

        // here its not necessary to update leaves_dir
        // since next_gen follows
        for (i=0; i<num_leaves; i++)
                if (l_list.Has_No_Kids(leaves_dir[i]) )
                        Prune_Blind_Branch_From_Leaf(i);

        return;
        }

template<class T>
inline void WellTree<T>::Prune_Blind_Branch_From_Leaf(p_leaf leaf)
        {
        Assert(leaves_dir.Dim()>=1);

        l_list.Destroy_Branch(leaves_dir[leaf]);

        return;
        }


//Backtrack_from(a_node) returns a new list with every element
//containing address of every nodes along path sequence
template<class T>
inline void WellTree<T>::Backtrack_From(ImpObjectList<T> & sequence,p_leaf leaf)
        {
        l_list.Backtrack_From(sequence,leaves_dir[leaf]);
        }

//start the next generation transform kid_dir leaves_dir;
template<class T>
inline void WellTree<T>::Next_Gen() // leaves=kid
        {
        leaves_dir=kid_dir;
        kid_dir.Reset();
        return;
        }

template<class T>
inline void WellTree<T>::Subst_Old_Kid_Destroy_Old_Branch_Ins_New(p_kid
old_kid,
                                        const T & info,p_leaf new_father)
        {
        l_list.Destroy_Node(kid_dir[old_kid]);
        kid_dir[old_kid]=l_list.Create(info,leaves_dir[new_father]);
```

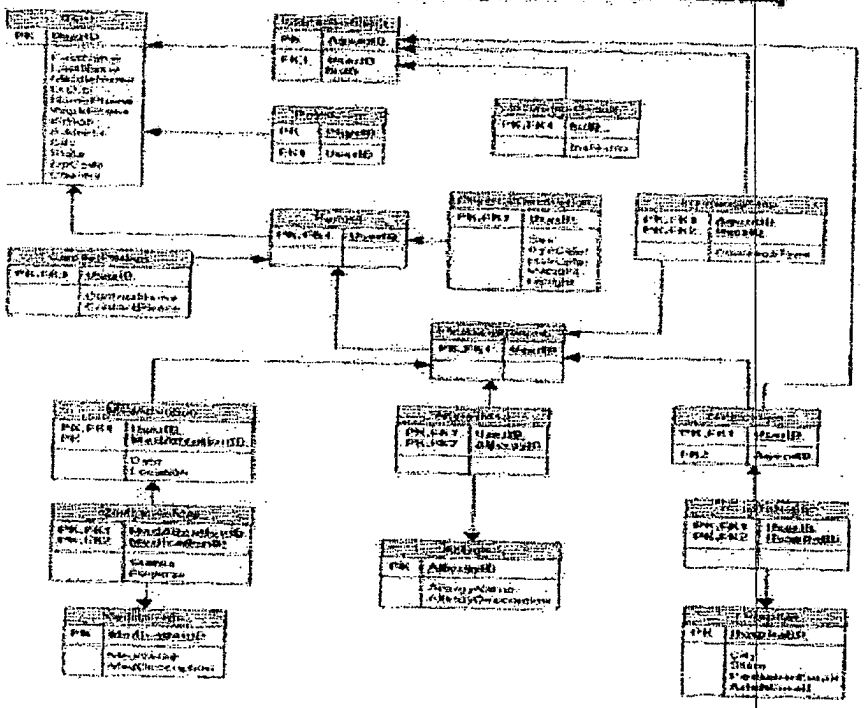
```
        return;
        }

#endif
```

APPENDIX F

AppendixB.jpg (JPEG Image, 720x540 pixels) - Scaled (82%)



[The structure below shows how tables in the database interact for data storage and retrieval for use by healthcare professionals and insurers.]



AppendixD.jpg (JPEG image, 720x540 pixels) - Scaled (82%)